GATE IT 2006 — Question: 50

The Code

```
#include <stdio.h>

void swap (int *x, int *y)
{
    static int *temp;
    temp = x;
    x = y;
    y = temp;
}

void printab ()
{
    static int i, a = -3, b = -6;
    i = 0;
    while (i <= 4)
    {
        if ((i++)%2 == 1) continue;
        a = a + i;
        b = b + i;
    }
    swap (&a, &b);
    printf("a =  %d, b = %d\n", a, b);
}

int main()
{
    printab();
    printab();
    return 0;
}
```

Step-by-Step Analysis

Function: `swap(int *x, int *y)`

- This function only swaps the local copies of the pointers. It never changes the values at the addresses (i.e. `*x` and `*y`). As a result, the intended swap of the integers does *not* occur.

Function: `printab()`

- **Declaration:**

```
static int i, a = -3, b = -6;
```

  Here, `a` and `b` being declared `static` are initialized only once (at the first call) and persist between calls. Also *note* that assignment resets the value of i to 0 every time the function is called, even though i is `static`.

- **Inside `printab()`:**

  1. `i = 0;` resets `i` to zero at the beginning of each call.
  2. `while (i <= 4)` starts a loop that will run as long as `i` is at most 4.
  3. Inside the loop:

```
if ((i++) % 2 == 1) continue;
a = a + i;
b = b + i;
```

     – The expression `(i++)` uses the current value of `i` then increments it.
     – If the original value of `i` is odd (i.e. %2 == 1), the `continue` statement skips the rest of the loop body.
     – Otherwise, the updated value of `i` is added to both `a` and `b`.

- **The while-loop works as follows (for each call):**

  First call to `printab()`:
  Initially: $a = -3$, $b = -6$, and $i = 0$.

Iteration	**Value of i (before `i++`)**	**i after `i++`**	**Condition**	**Updates to a and b**
1	0	1	$0\%2 = 0$ (false)	$a = -3 + 1 = -2$, $b = -6 + 1 = -5$
2	1	2	$1\%2 = 1$ (true)	No update (continue)
3	2	3	$2\%2 = 0$ (false)	$a = -2 + 3 = 1$, $b = -5 + 3 = -2$
4	3	4	$3\%2 = 1$ (true)	No update (continue)
5	4	5	$4\%2 = 0$ (false)	$a = 1 + 5 = 6$, $b = -2 + 5 = 3$

  The loop ends when $i = 5$ (since $5 \leq 4$ is false). Thus, after the loop:

$$a = 6, \quad b = 3.$$

- Next, `swap(&a,&b);` is called. However, as explained earlier, this call does not change the values of `a` and `b`.

- Finally, `printf("a = %d, b = %d", a, b);` prints:

$$a = 6, \quad b = 3.$$

Back to `main()`

```
int main()
{
    printab();
    printab();
    return 0;
}
```

- The first call to `printab()` (as shown above) prints:

$$a = 6, \quad b = 3.$$

- Since `a` and `b` are static in `printab()`, their updated values persist.
- For the second call, we start with the current values $a = 6$ and $b = 3$ and reset `i = 0` again.

  While-loop for the second call:

Iteration	i **(before `i++`)**	i **after `i++`**	**Condition**	**Updates**
1	0	1	$0\%2 = 0$ (false)	$a = 6 + 1 = 7,\ b = 3 + 1 = 4$
2	1	2	$1\%2 = 1$ (true)	No update
3	2	3	$2\%2 = 0$ (false)	$a = 7 + 3 = 10,\ b = 4 + 3 = 7$
4	3	4	$3\%2 = 1$ (true)	No update
5	4	5	$4\%2 = 0$ (false)	$a = 10 + 5 = 15,\ b = 7 + 5 = 12$

  After the loop, we have:

$$a = 15, \quad b = 12.$$

  Again, the call to `swap(&a, &b);` does not change these values.

- Finally, the second call prints:

$$a = 15, \quad b = 12.$$

Final Output

The program prints the following two lines:

```
a = 6, b = 3
a = 15, b = 12
```